OPIC
OFFICE DE LA PROPRIÉTÉ INTELLECTUELLE DU CANADA

CIPO
CANADIAN INTELLECTUAL PROPERTY OFFICE

(12) (19) (CA) **Brevet-Patent**

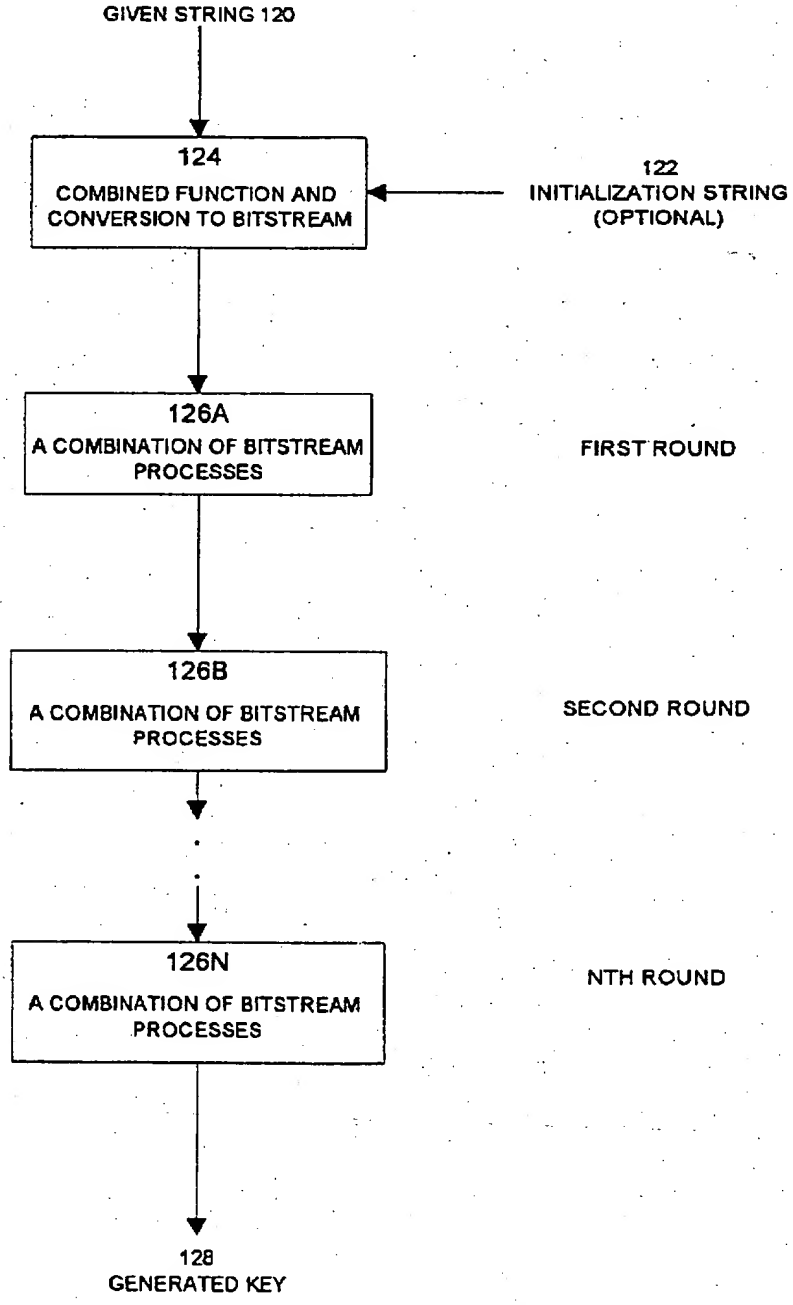
(11) (21) (C) **2,210,763**
(22) 1997/07/17
(43) 1999/01/17
(45) 2000/02/29

(72) KOU, Weidong, CA
(73) IBM CANADA LTD. - IBM CANADA LIMITÉE, CA

(51) Int.Cl.[6] H04L 9/32

(54) **GENERATION DE CLES D'AUTHENTIFICATION A PARTIR D'UNE CHAINE DE CARACTERES DONNEE**

(54) **KEY GENERATION FROM A GIVEN STRING FOR ENTITY AUTHENTICATION**



OPIC
OFFICE DE LA PROPRIÉTÉ INTELLECTUELLE DU CANADA

CIPO
CANADIAN INTELLECTUAL PROPERTY OFFICE

(11) (21) (C) **2,210,763**
(22) 1997/07/17
(43) 1999/01/17
(45) 2000/02/29

(57) Méthode de production d'une clé de codage symétrique pour une chaîne de caractères donnée, p. ex. mot de passe ou chaîne alphanumérique. Il n'y a pas de limite à la longueur ou au contenu de la chaîne. Celle-ci est convertie en un train de bits au moyen d'une méthode de codage de caractères normalisée, p. ex. ASCII. Le train de données fait ensuite l'objet de multiples séries de combinaisons d'opérations de traitement. Ces opérations peuvent comprendre ce qui suit : séparer le train de bits en de multiples sous-trains, réordonner et remanier les sous-trains, exécuter des opérations au niveau du bit logique sur les sous-trains, exécuter le hachage unilatéral des sous-trains et recombiner les multiples sous-trains en un train unique au moyen d'opérations au niveau du bit logique sur les sous-trains. Avant de convertir la chaîne de caractères en un train de bits, la chaîne de caractères donnée peut être optionnellement combinée à une chaîne de caractères d'initialisation particulière (par exemple, une chaîne d'initialisation associée à un processeur particulier), et c'est la chaîne de caractères combinée qui est ensuite codée et utilisée pour générer une clé de cryptage. L'application des multiples séries de combinaisons précitées, p. ex. seize, permet d'obtenir une clé de codage robuste.

(57) The invention provides a scheme for generating a symmetric encryption key for a given character string, such as a password or an alphanumeric. There is no limitation on the length or content of the given character string. The character string is converted to a bitstream using a standard character encoding scheme, such as ASCII. The bitstream is then subjected to combinations of bitstream processing operations over multiple rounds. The bitstream processing operations can include splitting the bitstream into multiple sub-bitstreams, reordering and shuffling the sub-bitstreams, performing logical bit-wise operations on the sub-bitstreams, performing one-way hashing on the sub-bitstreams, and combining the multiple sub-bitstreams back into a single bitstream through logical bit-wise operations on the sub-bitstreams. Prior to converting the character string into a bitstream, the given character string may be optionally combined with an initialization character string (for example, an initialization string associated with a particular processor), and it is the combined character string that is then encoded and used for generating an encryption key. Applying the combination rounds of the bitstream processes multiple times, for example sixteen times, will result in a strong key for encryption.

CA9-97-023

KEY GENERATION FROM A GIVEN STRING FOR ENTITY AUTHENTICATION

Abstract of the Disclosure

The invention provides a scheme for generating a symmetric encryption key for a given character string, such as a password or an alphanumeric. There is no limitation on the length or content of the given character string. The character string is converted to a bitstream using a standard character encoding scheme, such as ASCII. The bitstream is then subjected to combinations of bitstream processing operations over multiple rounds. The bitstream processing operations can include splitting the bitstream into multiple sub-bitstreams, reordering and shuffling the sub-bitstreams, performing logical bit-wise operations on the sub-bitstreams, performing one-way hashing on the sub-bitstreams, and combining the multiple sub-bitstreams back into a single bitstream through logical bit-wise operations on the sub-bitstreams. Prior to converting the character string into a bitstream, the given character string may be optionally combined with an initialization character string (for example, an initialization string associated with a particular processor), and it is the combined character string that is then encoded and used for generating an encryption key. Applying the combination rounds of the bitstream processes multiple times, for example sixteen times, will result in a strong key for encryption.

CA9-97-023

KEY GENERATION FROM A GIVEN STRING FOR ENTITY AUTHENTICATION

Field of the Invention

This invention generally relates to the field of networked communications and provides, in particular, a scheme for generating a symmetric encryption key from a given string, such as a password, which can be used for network entity authentication.

Background of the Invention

Many large and/or costly applications and databases reside on servers to which users have network access, either through an intranet (a contained network with a limited number of users) or by the Internet. Simply because a server or an application on a server is networked, does not necessarily mean that all users have equal access to the functions provided, or even that all users have access.

Controlled access is usually handled through authenticating the credentials of the user seeking access. U.S. Patent number 5,491,752, entitled System For Increasing The Difficulty Of Password Guessing Attacks In A Distributed Authentication Scheme Employing Authentication Tokens, of Digital Equipment Corp., describes one way in which this can be done. The transmission code between the sender and receiver is generated using a hashing algorithm. Both the sender and receiver have a common list of passwords and tokens, and the list is used to determine the particular hashing technique applied to the transmission code, and thus authenticate the user. However, having multiple passwords and tokens for each user is unwieldy in a large network.

The more common approach is to provide each user with a unique entity authentication key that has associated with it at the server the user's unique credentials for that server or application.

Maintaining the secrecy of the user's entity authentication key, particularly during its transmission over the network, is critically important to avoid compromising the security of the whole system. One way to achieve this is to encrypt the user's key prior to transmission.

There are many known encryption algorithms, such as DES and RSA.

DES (Data Encryption Standard) is a symmetric algorithm adopted as a federal standard in the late 1970's and the beginning of the 1980's. DES is a block-cipher which transforms data from 64 bit plain-data blocks into 64 bit cipher-data blocks. The key length of DES is 64 bits, of which 8 bits are used for parity checking and 56 bits are actual key data for encryption or decryption.

RSA is a public key (asymmetric) cryptographic algorithm named after its three inventors, Ron Rivest, Adi Shamir and Leonard Adleman. It is a public-key cryptosystem that can be used for both encryption and for digital signature. The security foundation of RSA is built on the difficulty of factoring large numbers; the public and private keys of RSA are functions of a pair of large prime numbers with a typical size of 512 to 2048 in bit length.

U.S. Patent No. 5,483,598, titled Message Encryption Using A Hash Function, of Digital Equipment Corp., discusses an encryption method which uses a key and XORes it with a block of a message. The key is produced by hashing a number and the message block using the hashing algorithm MD4 (Message Digest algorithm). A new key will be formed by hashing the previous key and the previous encrypted message.

However, encryption keys are not easy for human users to memorize. Thus, providing a way for the user to obtain an encryption key, for example using a known encryption method, is useful.

In U.S. Patent No. 5,073,935, titled Method For Secure Communication, of Jose Pastor, a set of numbers is provided that have the property that the encrypted version of the numbers will be of an order substantially smaller than that of the original set when the RSA public key encryption is used. To encrypt a message, a number is selected from the set and the selected number is used to produce a key by a hashing function. The requirement for the hashing function is that it maps a number a onto a set of numbers of lower order, such that the jth number from the number set may be used for an

encryption key. The key is then used to encrypt the message via the DES algorithm. The key itself is encrypted using the RSA algorithm. However, like the earlier discussed Patent No. 5,491,752, the requirement for the number set described in this system prevents its applicability for wide networks.

Summary Of The Invention

The easiest types of unique keys for human users to remember are passwords and simple alphanumeric codes. Therefore, the present invention is directed to providing a mechanism to generate encrypted keys from a given character string such as a password or alphanumeric, without limitation on string length.

The invention is also directed to a generic crypto-logic design which generates a high quality key that is highly random within the key space, so that security can be provided when using the key in an entity authentication process.

Accordingly, the present invention provides a process for generating an encryption key from a bitstream for a given character string. The process consists of applying, at least repetitively, a combination of at least two bitstream processing operations selected from the group of operations including bitstream splitting, bitstream reordering and shuffling, bit-wise operations, and one-way hashing. Preferably, different combinations of the bitstream processing operations are applied in subsequent rounds.

In a further embodiment, the invention provides a process for generating an encryption key from a given character string in which the given character string is first combined with an initialization character string to generate a combined character string. Character encoding is then applied to the combined character string to generate a first bitstream. In a first round, the bitstream is split into multiple sub-bitstreams and then the multiple sub-bitstreams are recombined into a second single bitstream on applying at least one additional bitstream processing operation selected from the group

of operations comprising reordering and shuffling the sub-bitstreams, bit-wise operations on the sub-bitstreams, one-way hashing on the sub-bitstreams and combining multiple sub-bitstreams into a single bitstream through bit-wise operations on sub-bitstreams. In at least an additional round, further bitstream processing operations are performed on the resultant single bitstream.

The invention also provides a mechanism for generating an encryption key, for use in network entity authentication, from a given character string. The mechanism includes means for converting the given character string into a bitstream and means for generating a random bitstream. The means for generating a random bitstream operate by repetitively applying a combination of at least two bitstream processes selected from a group of bitstream processes. These bitstream processes include splitting a bitstream into multiple sub-bitstreams, reordering and shuffling sub-bitstreams, performing bit-wise operations on sub-bitstreams, one-way hashing, and combining multiple sub-bitstreams into a single bitstreams. Preferably, the mechanism includes means for combining the given character string with an initialization character string to generate a combined character string, in which case the means for converting the given character string consists of means for converting the combined character string into a bitstream.

The invention also includes a program storage device having embodied thereon computer readable program code means to program a computer to perform the processes described above.

Brief Description of the Drawings

Embodiments of the invention will now be described in detail in association with the accompanying drawings, in which :

Figures 1 to 5 are schematic diagrams with accompanying flow diagrammatic examples illustrating several bitstream processes used, in combination, in aspects of the preferred embodiment of the present invention;

Figure 6 is a data flow diagram schematically illustrating a generic encryption key generation

mechanism combining processes from Figures 1 to 5; and

Figure 7 is a data flow diagram schematically illustrating a specific implementation of the encryption key generation mechanism of the invention.

Detailed Description of the Preferred Embodiments

The present invention provides a technique for generating an encrypted key from a given character string, such as a password or alphanumeric string. The crypto logic for a generic key generation scheme, according to one aspect of the invention, is based on multiple rounds of combinations of the bitstream processes illustrated in Figures 1 through 5. These processes include bitstream split (Figure 1), reordering and shuffling (Figure 2), bit-wise operations (Figure 3) and one way hashing (Figure 4).

The given character string, then, must first be converted into a bitstream by encoding the characters by a number of bits. The techniques to do this are well known to those skilled in the art. In North America, the characters are usually encoded by ASCII code in which each character is encoded by eight bits. The international standard, Unicode, encodes each character to sixteen bits which is able to represent the spoken text of many alphabets. For example, it can handle ideographic languages such as Chinese and Japanese.

Figure 5 illustrates an optional combination function of a given string and an initialization string employed prior to encoding the character string to a bitstream.

Referring first to Figure 1, bitstream split 4 is a technique that splits a bitstream 2 into multiple sub-bitstreams A, M and N, all generally designated by 6. The split can be achieved in any number of ways. Two examples are illustrated in the flow diagrams of Figures 1A and 1B.

Figure 1A illustrates the most straightforward form of bitstream splitting. Bits are assigned, one-by-one, to the sub-bitstreams in order, starting with the first bit assigned to the first substream (block

10), the second bit assigned to the second substream (block 14), and on to the Nth bit assigned to the Nth bitstream (block 18). If after any assignment the original bitstream has been exhausted, processing ends (blocks 12, 16 and 20). However, if bits remain in the original bitstream after assignment of the Nth bit to the Nth substream (blocks 18, 20), the process loops as the next bit is assigned to the first substream (block 22). The process is recursive until the original bitstream is exhausted. Taking a simple bitstream 010001 and splitting it into three substreams following the process illustrated in Figure 1A, would result in the following:

substream 1: 00
substream 2: 10
substream 3: 01

A simple variation on the process illustrated in Figure 1A is to assign two bits at a time from the original bitstream to the substreams in order. Under this variation, the three substreams derived from the original bitstream 010001 would be:

substream 1: 01
substream 2: 00
substream 3: 01

Figure 1B illustrates another example of bitstream splitting that is more complicated, in that different numbers of bits from the original bitstream are assigned to successive sub-bitstreams. After assigning the first two bits from the original bitstream to the first substream (block 30), only one bit is assigned to the next substream (block 36) and three bits to the substream after that (block 52). As with the previously described algorithm (Figure 1A), processing is handled recursively (blocks 40, 54) and ends when the original bitstream has been exhausted (blocks 32, 42-44, 48-50, 56-58 and 62-66). Using the original bitstream 010001, the three sub-bitstreams resulting from use of the method in Figure 1B would be:

substream 1: 01
substream 2: 0
substream 3: 001

Bitstream reordering and shuffling 72, illustrated schematically in Figure 2, removes the correlation

among bits within a bitstream 70 and so that the resulting bitstream 74 is random. For example, a 10-bitstream could be reordered to place the bits in the following order:

10 7 8 4 5 3 9 1 2.

Bit-wise operations 78 (Figure 3) combine M bitstreams 76 or sub-bitstreams into L bitstreams 80 using combinations of the logical operations AND, OR and Exclusive-OR (XOR).

These three operations are conventionally defined as follows:

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

XOR	0	1
0	0	1
1	1	0

Depending on the combination of these operations, the number of streams output may be the same, or greater or less than the number input. A simple example is illustrated schematically in figure 3A. Two input streams 76a are combined three time 78a using each of the logical operations, AND, OR and XOR, to produce three output streams 80a. Thus, if:

input stream 1 = 0101

input stream 2 = 1111

then,

output stream 1 = 0101

output stream 2 = 1111

output stream 3 = 1010

The one way hashing process 84 illustrated in Figure 4 is irreversible. It produces a hashed bitstream 86 from the original bitstream 82. From the one way hashing process, it is computationally infeasible either to produce the original bitstream, or to find two different bitstreams which produce the same hashed bitstream. In the preferred embodiment, the message digest algorithm MD5 is used, as discussed below. Other known algorithms include the MD4 and MD2 message digest algorithms, and the standard, Secure Hash Algorithm (SHA).

A given string be optionally combined with an initialization string tied to the address or physical location of a computer, as illustrated in Figure 5. This technique is particularly useful in a closed network, where access limited to select machines, as well as select users, is desired. Figure 5 illustrates the basic concept of combining 92 a string 88 given by a user with an initialization string 90 supplied at the machine used to produce a single combined string.

The method of combination can vary. Figure 5A illustrates one possible algorithm. When a user entered character string is received 100, an initialization character string is loaded 102. The combination of the two strings is done by taking one character at a time from the first string 104 then one character at a time from the second string 106, and by repeating this process until all characters in both strings are taken 108, 114. If the length of the two strings are not equal, then after taking the last character from the shorter string, the remaining character from the longer string will be appended to the combined string, that is after taking a character from a string the mechanism simply tests whether there are any additional characters in the other string 110, 116. If there are not additional characters in the second string when it is tested 110, then the remaining characters from the first string are simply appended 112, and if there are not additional characters remaining in the first string 116, then the remaining characters from the second string are simply appended 118.

An example of the result of this process is as follows. Taking two strings, "abcdefghijklmn" and "mypassword", the combined string "ambycpdaesfsgwhoirjdklmn" is formed.

Figure 6 illustrates the combination of the above mentioned bit stream processes in a generic encryption key generation mechanism, according to the invention.

A given string 120 can be optionally combined with an initialization string 122 and encoded to a bitstream as the first step through a combination and encoding function 124. The combination function performs a string shuffling to form the input string to the generic key generation scheme, that essentially comprises multiple rounds of combinations of the bit stream processes illustrated in Figures

1 to 4 and discussed above. Depending on the encryption mechanism (eg., DES), the generated key 128 from Figure 6 may require a post process such as parity bit setting.

Figure 7 provides an implementation example of a specific key generation mechanism following the generic outline illustrated in Figure 6. Figure 7 illustrates a mechanism to generate a 64 bit DES key from a given string for authentication purpose.

The initialization character string 200 is used for scambling the user entered string (a password or alphanumeric) 202. This is performed in the combination block 204, using the process such as illustrated in Figure 5A and described above. The bitstream generated from the combined string is hashed by the MD5 message digest algorithm 206 to yield a 16 byte digest. The hashed result is then divided into two halves, a left half 208 and right half 212. Each half is hashed again by the MD5 message digest algorithm 210, 214. The result from hashing the right half 212 is bitwise reversed after hashing 214. (For example, "0011001100110011" becomes "1100110011001100"). The results from hashing the left half 210 is exclusive ORed 216 with the bit reversed right half 214 to form the input for the second round process. The process is repeated n times 218 through 230, 234. Finally, the exclusive ORed result 236 is divided again, and this time the right half 240 is exclusive ORed 242 with the left half 238. At this point, 8 byte data is produced, which forms a 64 bit key. Since the DES encryption key has parity checking bits, the final DES encryption key 246 is produced after setting the parity bits with the 8 byte data 244.

The generic key generation method of the present invention could be provided as a specific implementation (such as shown and described in relation to Figure 7), or as a tool for application developers to design a key for unique encryption "algorithms" for specific applications or specific users. Provided a minimum number of rounds of combined bitstream operations are performed in the algorithm, the resulting key for encryption will be strong enough for secure network transmission. A strong key will be achieved if 16 rounds of combined bitstream operations are applied. A lesser number of rounds may be adequate where the security requirements are not as stringent.

A feature of the invention discussed above is that the specific algorithm produced following the method of the invention will not limit the user's password as to size or context. "Hereismypassword" will be equally valid password input for key generation as "pas8word" or some other rigidly-structured password input. With this feature, the user can use a long English phrase mixed with digits, against password guessing attack, such as "My grandfather's 90th birthday is March 16".

Modifications to the invention which would be obvious to those skilled in the art are intended to be covered by the appended claims.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. A process for generating an encryption key from a bitstream for a given character string, comprising:

applying, at least repetitively, a combination of at least two bitstream processing operations selected from the group of operations including bitstream splitting, bitstream reordering and shuffling, bit-wise operations, and one-way hashing.

2. The process, according to claim 1, wherein in repetitive applications, different combinations of the at least two bitstream processing operations are applied.

3. The process, according to claim 2, wherein combinations of at least two bitstream processing operations are applied at least eight times.

4. The process, according to claim 2, wherein combinations of at least two bitstream processing operations are applied at least sixteen times.

5. The process, according to claim 1, wherein the combination of at least two bitstream processing operations comprises:

splitting the bitstream into multiple sub-bitstreams; and

applying bit-wise operations to combine the multiple sub-bitstreams into a second single bitstream.

6. The process, according to claim 5, wherein the combination of at least two bitstream processing operations comprises:

applying to at least one of the sub-bitstreams, at least one bitstream processing operation selected from the group of operations including bitstream reordering and shuffling, bit-wise operations

and one-way hashing.

7. A process for generating an encryption key from a given character string, comprising:

combining the given character string with an initialization character string to generate a combined character string;

applying character encoding to generate a first bitstream from the combined character string;

in a first round, splitting the first bitstream into multiple sub-bitstreams and recombining the multiple sub-bitstreams into a second single bitstream on applying at least one additional bitstream processing operation selected from the group of operations comprising reordering and shuffling the sub-bitstreams, bit-wise operations on the sub-bitstreams, one-way hashing on the sub-bitstreams and combining multiple sub-bitstreams into a single bitstream through bit-wise operations on sub-bitstreams; and

in at least an additional round, performing further bitstream processing operations on the resultant single bitstream.

8. The process for generating an encryption key, according to claim 7, wherein the step of, in at least an additional round, performing further bitstream processing operations on the resultant single bitstream, comprises performing combinations of bitstream processing operations in successive rounds.

9. The process, according to claim 8, wherein the successive rounds comprise at least seven additional rounds.

10. The process, according to claim 8, wherein the successive rounds comprise at least fifteen additional rounds.

11. A mechanism for generating an encryption key, for use in network entity authentication, from a given character string, comprising:

means for converting the given character string into a bitstream; and

means for generating a random bitstream by repetitively applying a combination of at least two bitstream processes selected from the group of bitstream processes comprising:

splitting a bitstream into multiple sub-bitstreams, reordering and shuffling sub-bitstreams, performing bit-wise operations on sub-bitstreams, one-way hashing, and combining multiple sub-bitstreams into a single bitstreams.

12. The mechanism, according to claim 11, further comprising:

means for combining the given character string with an initialization character string to generate a combined character string,

and wherein the means for converting the given character string comprises means for converting the combined character string into a bitstream.

13. The mechanism, according to claim 11, wherein the means for generating a random bitstream, comprises:

means for splitting the bitstream into multiple sub-bitstreams;

means for applying at least one further bitstream processing operation to at least one of the sub-bitstream; and

means for recombining the multiple sub-bitstreams into a second single bitstream.

14. The mechanism, according to claim 13, wherein the means for recombining the multiple sub-bitstreams comprises means for combining the sub-bitstreams into a single bitstream through bit-wise operations.

15. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for generating an encryption key from a bitstream for a given character string, said method steps comprising:

applying, at least repetitively, a combination of at least two bitstream processing operations selected from the group of operations including bitstream splitting, bitstream reordering and shuffling, bit-wise operations, and one-way hashing.

16. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for generating an encryption key from a given character string, said method steps comprising:

combining the given character string with an initialization character string to generate a combined character string;

applying character encoding to generate a first bitstream from the combined character string;

in a first round, splitting the first bitstream into multiple sub-bitstreams and recombining the multiple sub-bitstreams into a second single bitstream on applying at least one additional bitstream processing operation selected from the group of operations comprising reordering and shuffling the sub-bitstreams, bit-wise operations on the sub-bitstreams, one-way hashing on the sub-bitstreams and combining multiple sub-bitstreams into a single bitstream through bit-wise operations on sub-bitstreams; and

in at least an additional round, performing further bitstream processing operations on the resultant single bitstream.

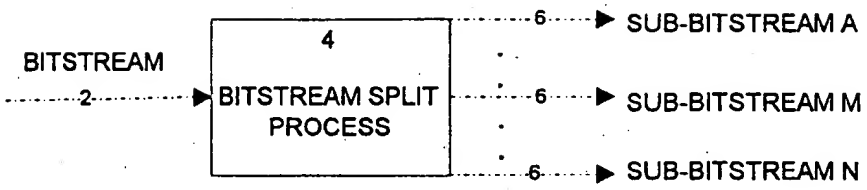

BITSTREAM
2
4
BITSTREAM SPLIT PROCESS
6
SUB-BITSTREAM A
6
SUB-BITSTREAM M
6
SUB-BITSTREAM N

FIGURE 1

FIGURE 1A

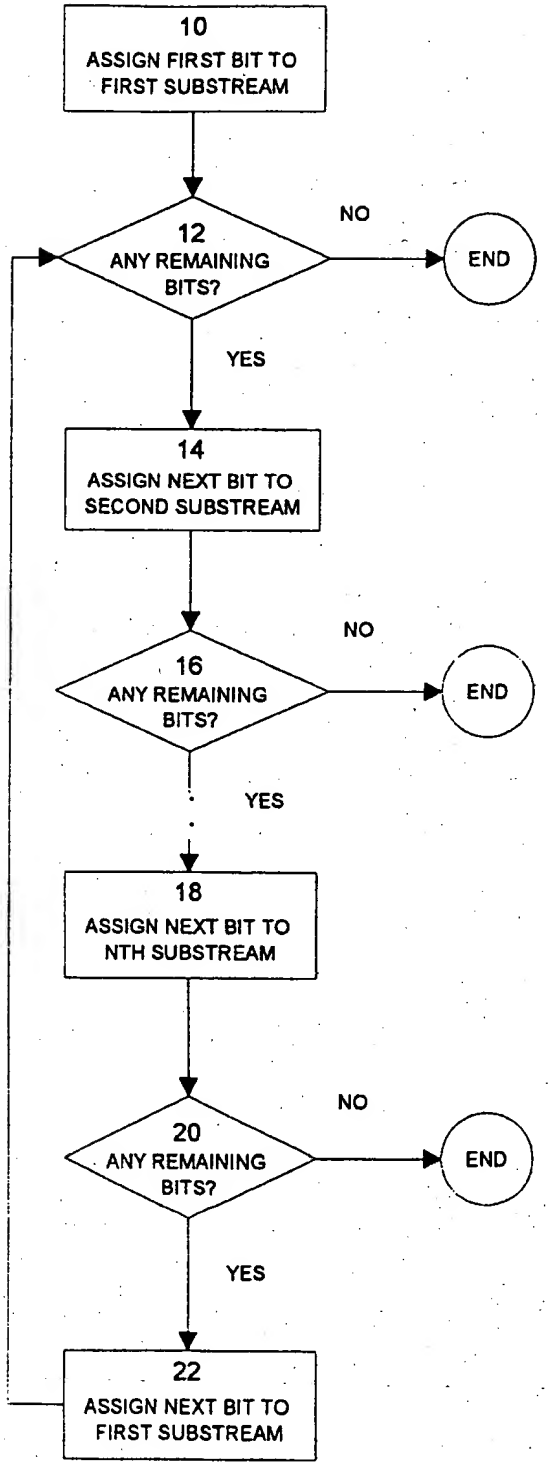

10
ASSIGN FIRST BIT TO FIRST SUBSTREAM
12
ANY REMAINING BITS?
NO
END
YES
14
ASSIGN NEXT BIT TO SECOND SUBSTREAM
16
ANY REMAINING BITS?
NO
END
YES
18
ASSIGN NEXT BIT TO NTH SUBSTREAM
20
ANY REMAINING BITS?
NO
END
YES
22
ASSIGN NEXT BIT TO FIRST SUBSTREAM

FIGURE 1B

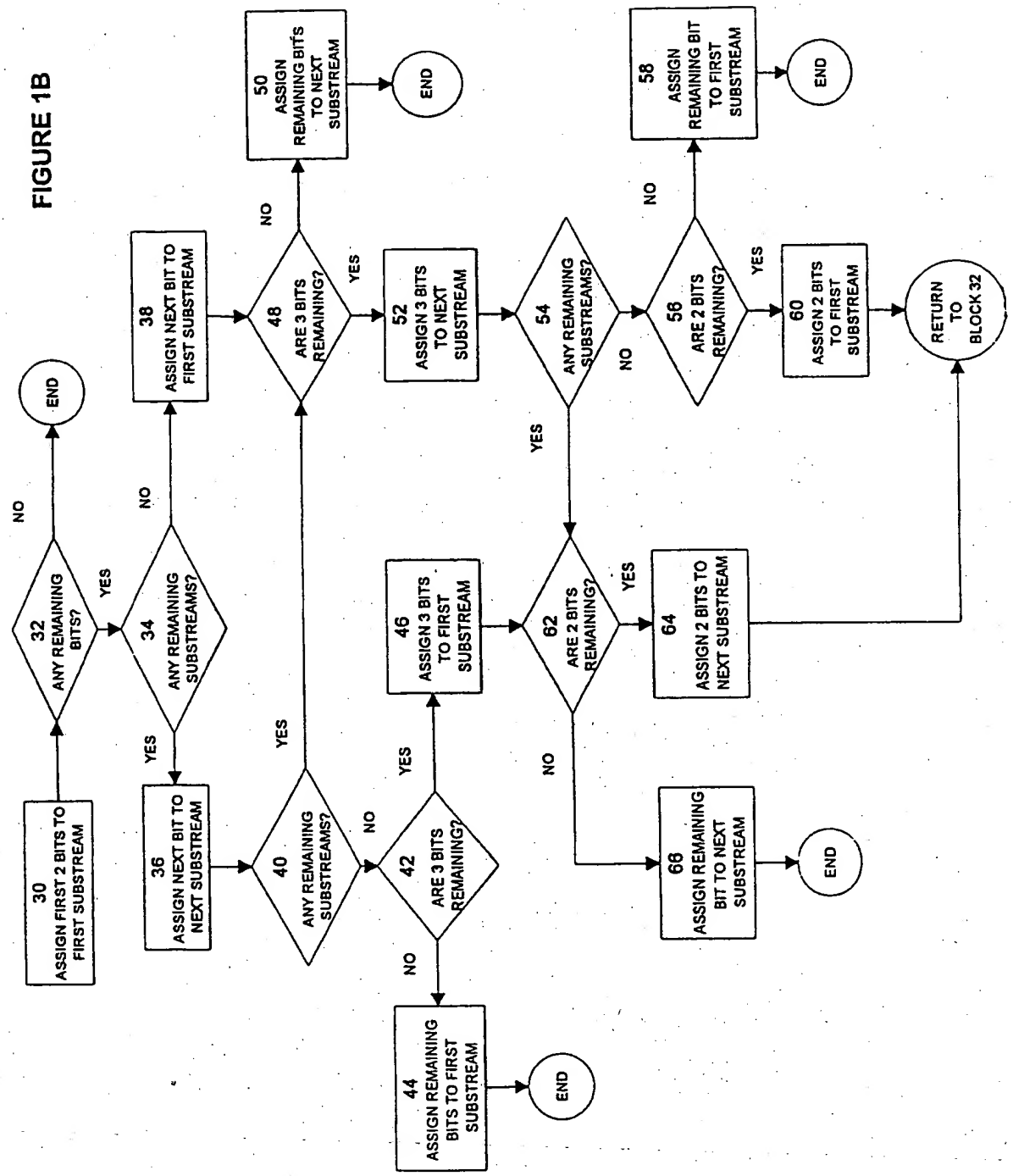

30
ASSIGN FIRST 2 BITS TO FIRST SUBSTREAM
32
ANY REMAINING BITS?
YES
NO
END
34
ANY REMAINING SUBSTREAMS?
YES
NO
36
ASSIGN NEXT BIT TO NEXT SUBSTREAM
38
ASSIGN NEXT BIT TO FIRST SUBSTREAM
40
ANY REMAINING SUBSTREAMS?
YES
NO
42
ARE 3 BITS REMAINING?
YES
NO
44
ASSIGN REMAINING BITS TO FIRST SUBSTREAM
END
46
ASSIGN 3 BITS TO FIRST SUBSTREAM
48
ARE 3 BITS REMAINING?
YES
NO
50
ASSIGN REMAINING BITS TO NEXT SUBSTREAM
END
52
ASSIGN 3 BITS TO NEXT SUBSTREAM
54
ANY REMAINING SUBSTREAMS?
YES
NO
56
ARE 2 BITS REMAINING?
YES
NO
58
ASSIGN REMAINING BIT TO FIRST SUBSTREAM
END
60
ASSIGN 2 BITS TO FIRST SUBSTREAM
62
ARE 2 BITS REMAINING?
YES
NO
64
ASSIGN 2 BITS TO NEXT SUBSTREAM
66
ASSIGN REMAINING BIT TO NEXT SUBSTREAM
END
RETURN TO BLOCK 32

FIGURE 2

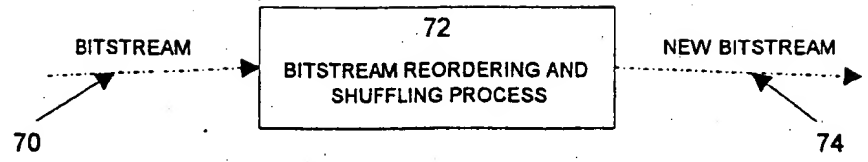


FIGURE 3

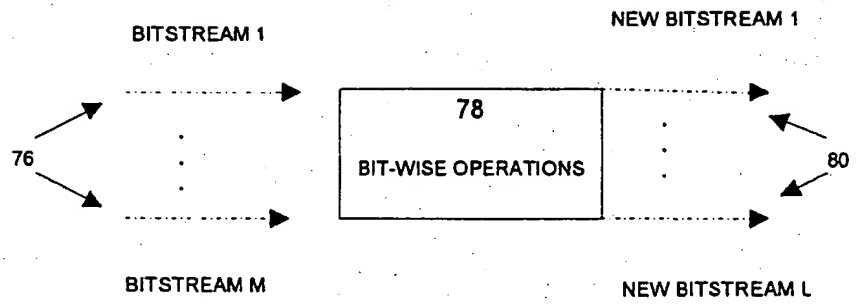


FIGURE 3A

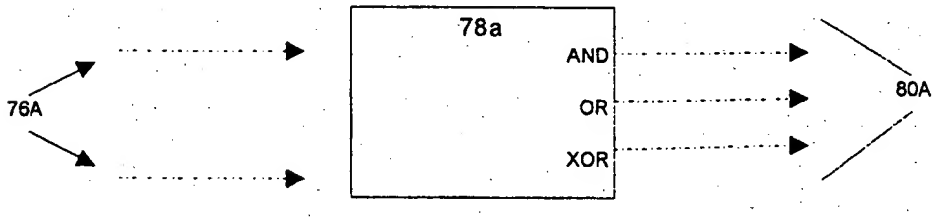


FIGURE 4

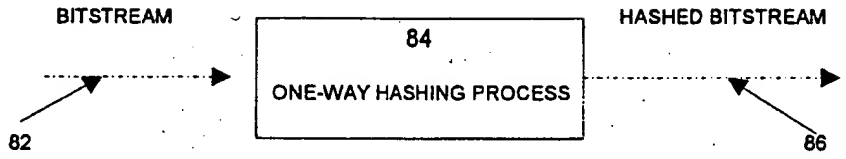


FIGURE 5

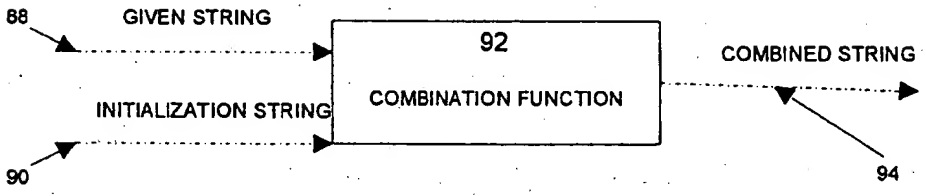


FIGURE 5A

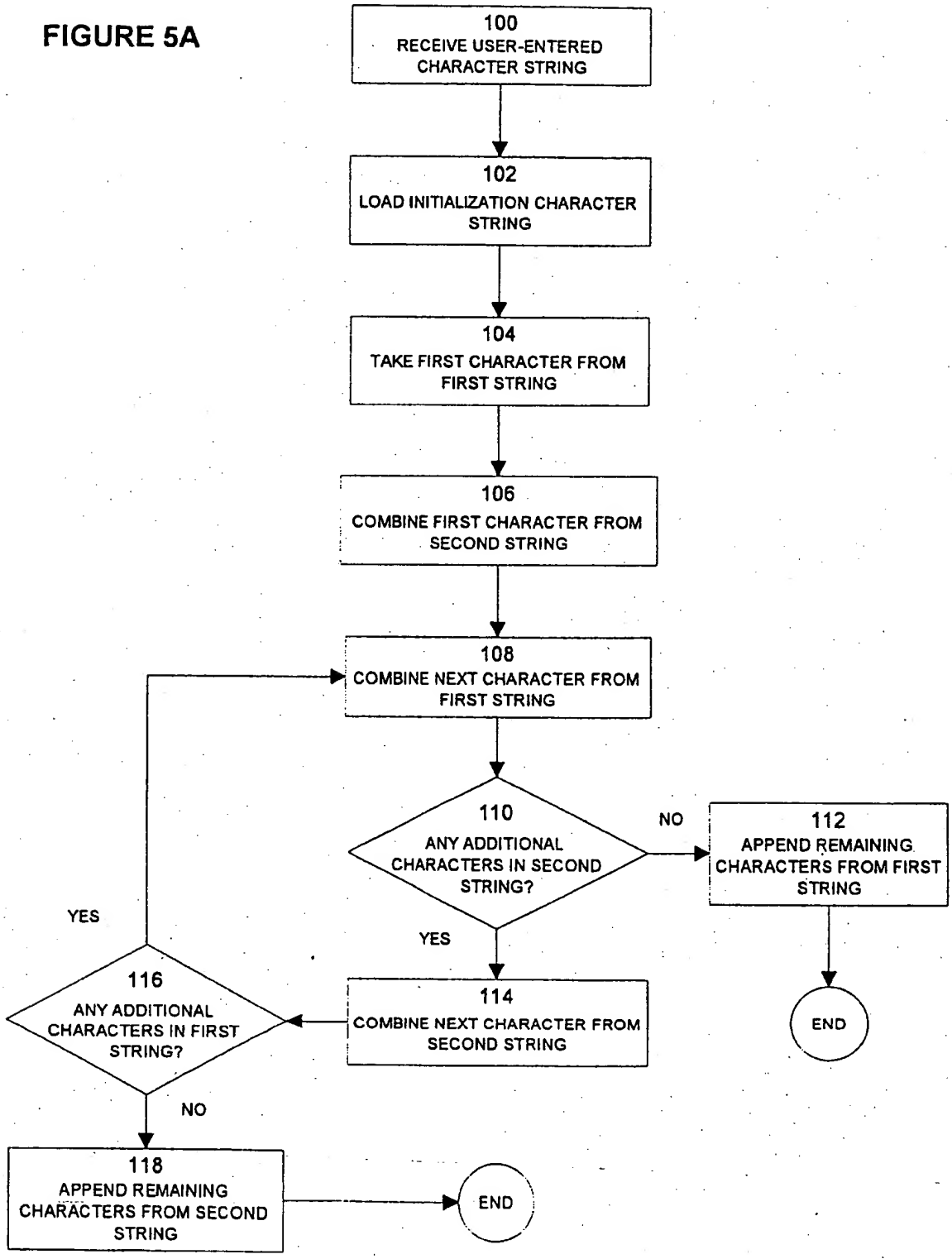
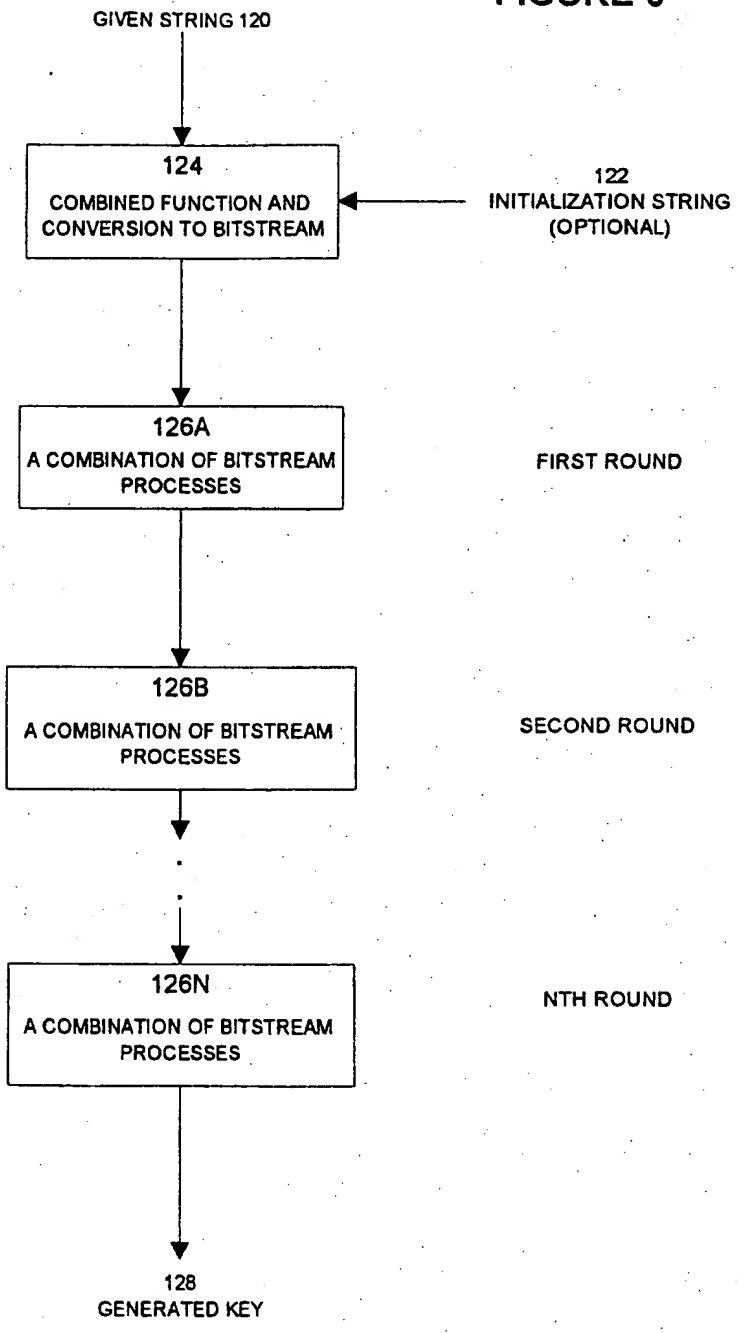

100
RECEIVE USER-ENTERED CHARACTER STRING
102
LOAD INITIALIZATION CHARACTER STRING
104
TAKE FIRST CHARACTER FROM FIRST STRING
106
COMBINE FIRST CHARACTER FROM SECOND STRING
108
COMBINE NEXT CHARACTER FROM FIRST STRING
110
ANY ADDITIONAL CHARACTERS IN SECOND STRING?
NO
112
APPEND REMAINING CHARACTERS FROM FIRST STRING
END
YES
114
COMBINE NEXT CHARACTER FROM SECOND STRING
116
ANY ADDITIONAL CHARACTERS IN FIRST STRING?
YES
NO
118
APPEND REMAINING CHARACTERS FROM SECOND STRING
END

FIGURE 6


GIVEN STRING 120
124
COMBINED FUNCTION AND CONVERSION TO BITSTREAM
122
INITIALIZATION STRING (OPTIONAL)
126A
A COMBINATION OF BITSTREAM PROCESSES
FIRST ROUND
126B
A COMBINATION OF BITSTREAM PROCESSES
SECOND ROUND
126N
A COMBINATION OF BITSTREAM PROCESSES
NTH ROUND
128
GENERATED KEY

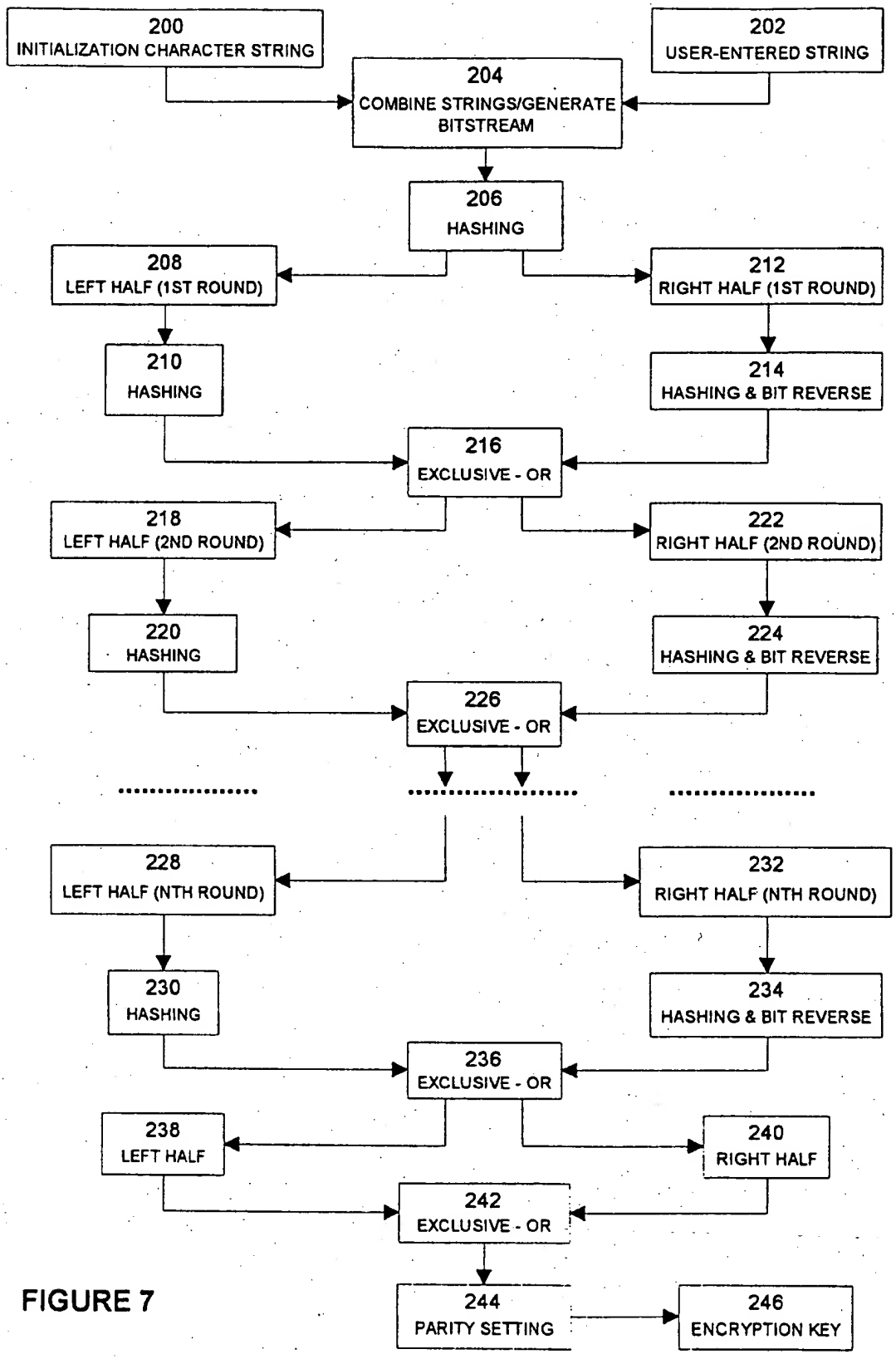

200
INITIALIZATION CHARACTER STRING
202
USER-ENTERED STRING
204
COMBINE STRINGS/GENERATE BITSTREAM
206
HASHING
208
LEFT HALF (1ST ROUND)
212
RIGHT HALF (1ST ROUND)
210
HASHING
214
HASHING & BIT REVERSE
216
EXCLUSIVE - OR
218
LEFT HALF (2ND ROUND)
222
RIGHT HALF (2ND ROUND)
220
HASHING
224
HASHING & BIT REVERSE
226
EXCLUSIVE - OR
228
LEFT HALF (NTH ROUND)
232
RIGHT HALF (NTH ROUND)
230
HASHING
234
HASHING & BIT REVERSE
236
EXCLUSIVE - OR
238
LEFT HALF
240
RIGHT HALF
242
EXCLUSIVE - OR
244
PARITY SETTING
246
ENCRYPTION KEY

FIGURE 7